Training and Awareness

Kenneth R. van Wyk, Software Engineering Institute [vita[3]]

2005-09-26; Updated 2008-08-28 L3 / M[4]

This article provides guidance on training and awareness initiatives in the field of software security. It examines the state of the practice of commercial software security training and awareness offerings and makes recommendations for goals and curricula contents.

Overview

This document is intended to provide guidance on training and awareness initiatives in the field of software security. It examines the state of the practice of commercial software security training and awareness offerings and makes recommendations for goals and curricula contents.

An effective training program is vital to adopting new software development practices. And, because software security is such an emerging field, there are very few experienced developers that are familiar with the sorts of practices described here in the Build Security In (BSI) portal. As such, a clearly defined training and awareness campaign is particularly important for this effort. Indeed, Microsoft considers developer training to be a vital component of its Security Development Lifecycle (SDL) [Howard 2006[16]], and makes annual training a requirement for all of its software engineers.

A commonly heard gripe in the industry is that academic curricula do not adequately address software security issues. We take a brief look here at curricula from the top universities in the United States and compare them against the training offerings found in the commercial sector.

When possible, it is a good idea to guide training programs by a recognized common body of knowledge (CBK). Although no such CBK for software security or software assurance appears to exist in the public space as of this writing, the U.S. Department of Homeland Security (DHS) is in the final stages of capturing and documenting one.

3. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/202-BSI.html (van Wyk, Ken)
5. #dsy256-BSI_over
6. #dsy256-BSI_sopcom
7. #dsy256-BSI_spoacad
8. #dsy256-BSI_bp
9. #dsy256-BSI_meas
10. #dsy256-BSI_buscase
11. #dsy256-BSI_gloss
12. #dsy256-BSI_train
13. #dsy256-BSI_curric
16. #dsy256-BSI_Howard

State of the Practice—Commercial

To assess the state of commercial training offerings available today, we examined the publicly available documentation, syllabuses, etc. from numerous commercial organizations. We looked for training that emphasizes as much of the BSI concepts as possible, including the best practice activities, knowledge base topics, and available tools. We also looked for training that is largely process agnostic, as are the concepts laid out in BSI.

The good news is that there are indeed many offerings to choose from. They range in size and scope, and they cover a broad spectrum of aspects of software security. The biggest strength in the available courses is that most of them provide a good amount of detail on the technical nature of current problems and available solution sets.

That said, the bad news is that many of the available courses appear to suffer from various shortcomings, at least with regard to the approaches presented in BSI. What follows is a brief description of those shortcomings, along with recommendations on how to avoid or alleviate them.

- Security software vs. software security. According to their syllabuses, many of the software security training offerings spend a great deal of time describing security functionality (the use of encryption, identification and authentication mechanisms, etc.). Although these security functions are vital ingredients of software security, they are essentially security ingredients. The topic of software security goes far beyond a simple list of security ingredients.
- Knowledge vs. practices. Similarly, many of the training offerings focus on individual problems (e.g., buffer overflows) and their respective point solutions. Avoiding these pitfalls is also vital to writing secure software, but much else needs to be covered for the training to be as effective as it needs to be.
- Network and operating system focus. Although not as common as the above two errors, some of the available training offerings appear to present a network and operating system centric point of view to the issues of developing secure software.

While all of the above elements are important to cover in a software security training program, what we feel is principally lacking in many of the course offerings is adequate coverage of security processes that are necessary to incorporate software security into the development processes and practices. These should be at least similar to those presented in the Best Practices area of BSI. Further, an emphasis should be placed on some high-value best practice activities such as abuse case[26] analysis, architectural risk analysis[27], risk-based testing[28], and code review practices[29] and tools[30].

At least the situation is steadily improving, with numerous high-quality courses becoming available recently. A sampling of these courses and the companies offering them is listed at the end of this article.

State of the Practice—Academic

Within academia, many top universities are now offering optional senior-level undergraduate and graduate courses in software security. These courses tend to be broader in focus than their commercial counterparts, in that they include discussions on the sorts of best practice activities mentioned above in addition to discussions of common vulnerabilities.

There is certainly room for optimism in these findings, while at the same time there is perhaps even more room for improvement. For example, a strong argument for integrating discussions of secure design processes, avoiding buffer overflows, and similar topics into the more general computer science courses could easily be made; why not teach students to avoid strcpy() and the like in an Introduction to C course?

26. http://buildsecurityin.us-cert.gov/bsi/resources/articles/series/bsi-ieee/125-BSI.html (Misuse and Abuse Cases: Getting Past the Positive)
27. http://buildsecurityin.us-cert.gov/bsi/articles/best-practices/architecture/10-BSI.html (Architectural Risk Analysis)
28. http://buildsecurityin.us-cert.gov/bsi/articles/best-practices/testing/255-BSI.html (Risk-Based and Functional Security Testing)
29. http://buildsecurityin.us-cert.gov/bsi/articles/best-practices/code/214-BSI.html (Code Analysis)
30. http://buildsecurityin.us-cert.gov/bsi/articles/tools/code.html (Source Code Analysis)

Integrating software security into the entire curriculum is bound to be more effective than offering it as a senior-level elective course.

Best Practices in Training and Awareness

As stated above, we feel that a best practice software security training program today should encompass the various best practices, knowledge bases, and tools presented in BSI. Further, training and awareness initiatives should plan for—at a minimum—three target audiences: senior decision makers, engineering managers, and software developers. Each of the audiences should receive training that addresses its needs, naturally. The most fundamental goals and objectives for each audience follow.

- Senior decision makers

  For a software security initiative to succeed in an organization, the organization's senior decision makers need to support the initiative with their buy-in. Therefore, an awareness training program should be presented to them that clearly articulates the need for software security and the difficulties faced in delivering secure software. The training should also succinctly describe the best practices necessary to accomplish those deliveries.
- Engineering managers

  Likewise, engineering or software development managers (across all of the organizations and disciplines involved in the overall development process) also need to buy into a software security initiative. Additionally, however, they need to have a thorough understanding of the software security practices that their organization will be incorporating into its development processes and specifically what their sub-organizations will need to do as a result. This is essential for managers to understand for purposes of project planning and execution. Thus, managers should be provided training content that describes the need for software security, as well as a thorough description and understanding of their organization's software security practices. The training should emphasize, where possible and feasible, the levels of effort for each software security activity involved, how to identify areas for improving existing software security practices (e.g., evaluate and improve), as well as methods for measuring a development organization's software security effectiveness. It may be beneficial to tailor the training content by audience somewhat—e.g., development managers are likely to have different specific needs and interests than test and quality assurance managers. In this case, the disciplines that each audience are directly responsible for should be emphasized and covered in more detail than the others.
- Software developers

  Software developers should receive training that provides them with a conceptual foundation of software security, its importance to their organization, and the practices used within their organization. They should gain practical knowledge about the benefits of software security. Additionally, developers should receive technology-specific security training in each of the technologies that are involved in designing, coding, and testing software in the technologies that they work with. In the same manner that the training for the management may be tailored by the audience's disciplines, the content for software developers must emphasize the aspects of software development that they work on directly. Further, it should be replete with code examples and hands-on exercises/labs to help reinforce the course material [Howard 2006[41]].

With these goals and objectives in mind, the following outlines are presented as guidelines for developing organizational curricula for software security training.

Training Courses and Outlines by Audience

For each of the three audiences, it is particularly useful to clearly address the rationale for each security activity. Where feasible, consider demonstrating the activity through exercises, examples, and in-depth anecdotes from case studies.

Software security awareness training for *senior decision makers* should look similar to the following:

1. Introduction to software security problems

This training module should present an overview of the security problems faced today by software developers. Its aim should be to convince the audience why traditional and largely separate approaches to information security and software development are flawed from a software security perspective. Further, it should present an accurate business case that weighs the often conflicting goals of development and security.

1. Shortcomings of traditional perimeter-based network security solutions
2. Common software weaknesses
3. Balancing the different goals of security and software development

2. Software security activities to integrate into the SDLC

   This module should provide a basic conceptual overview of software security activities and their impact on the SDLC for the senior decision maker audience. It should principally focus on describing the activities and their associated costs—monetary and schedule.

   1. Requirements and specifications activities
   2. Design time activities
   3. Implementation activities
   4. Test planning and testing
   5. Deployment, operations, and maintenance issues

Software security awareness training for *engineering management*, as stated, should be substantially similar to that provided to the senior decision makers, but with a somewhat different core message. Specifically, instead of solely aiming to convince the audience of the merits of software security, it should also ensure that the managers have the necessary knowledge to implement and measure an appropriate set of software security practices. Their training outline should look similar to the following:

1. Introduction to software security problems

   This training module should delve into the security problems faced today by software developers. Its aim should be to convince this management audience why traditional and largely separate approaches to information security and software development are flawed. Further, it should present an accurate business case that weighs the often conflicting goals of development and security. For the engineering management audience, particular attention should be paid to cost vs. benefit information, as they're often the people within an organization that are the most skeptical about the benefits of adding more activities to the SDLC process.

   1. Shortcomings of traditional perimeter-based network security solutions
   2. Common software weaknesses
   3. Balancing the different goals of security and software development

2. Software security activities to integrate into the SDLC

   This module should be the core of the training content provided to the engineering management audience. For the managers, particular focus should be given to describing the processes involved and how to implement them, as well as methods of measuring their teams' progress and successes. Additionally, realistic program management information should be provided, such as scheduling issues and typical level of effort required for each activity.

   1. Requirements and specifications activities
   2. Design time activities
   3. Implementation activities
   4. Test planning and testing
   5. Deployment, operations, and maintenance issues

Many of the same topics covered in the above two training curricula should also be covered for the *software developer* audience; however, the focus here should shift dramatically from the conceptual to the technical. A conceptual foundation must be presented, but the developers will need specific technical information in order for them to do their expected software security tasks. Additionally, where feasible and possible, hands-

on exercises should be incorporated into the training so that the developers can experiment with putting into practice the processes described in the training material.

1. Introduction to software security problems

   This training module should delve into the security problems faced today by software developers. Its aim, quite simply, should be to convince the students that they should care about software security in their work. (Realistic case studies can be highly beneficial here.)

   1. Shortcomings of traditional perimeter-based network security solutions
   2. Common software weaknesses
   3. Balancing the different goals of security and software development

2. Software security activities to integrate into the SDLC

   This module should present the same basic concepts that were presented to the engineering managers, but the principal focus should be on actionable recommendations for the developers. The students should come away with a clear understanding of where they fit into the software security program within their organization, what is expected of them, and how they need to implement software security. Specific guidance and recommendations should be provided for each of these processes that will help the student ”internalize” the correct behaviors as they apply to software security activities.

   1. Requirements and specifications activities
   2. Design time activities
   3. Implementation activities
   4. Test planning and testing
   5. Deployment, operations, and maintenance issues

3. Know the enemy

   To build software that can withstand attacks, it is essential to understand the nature of the anticipated attacks and the concepts behind them, and in considerable technical detail. This module should teach developers about their adversaries. The students should understand who wants to attack their software, why, and how they are likely to go about doing it. Common concepts such as buffer overflows, SQL injection, cross-site scripting, and so on should be thoroughly described and, where feasible, included in hands-on exercises/labs so that the students can best internalize the course material.

   1. Threat analysis – who are the attackers and what motivates them
   2. Common software vulnerabilities explained in detail – architectural flaws as well as implementation bugs
   3. Attack tools and methodologies

4. Knowledge base and tools

   Whereas the previous module stresses the best practices that developers are to follow, this module should arm the students with the necessary knowledge base and understanding of the tools necessary to their jobs. The actual content delivered here to each student may need to be further refined to the exact discipline of the student audience. For example, software designers need to focus on the architectural risk analysis processes and specific methodologies, whereas coders need to focus on code review methods and tools.

   1. Risk analysis techniques (e.g., STRIDE, SQM, CLASP)[91]
   2. Language-specific tips, pitfalls to avoid, rules, and guidelines
   3. Tools for code analysis, testing, etc.

5. Code remediation

   With the fundamentals out of the way, it is important to include training that includes prescriptive information on how to design and implement safe software. The content should be replete with code examples and should be specific to the languages, frameworks, etc., in use by the developers. At a minimum, the courseware should include example design and code patterns addressing the most

egregious bugs and flaws found in similar architectures and languages. It should include the OWASP Top 10[94] for web developers, for example.

1. Top 10 security defects (by technology)
2. Safe coding examples and guidelines
3. Exercises/labs to reinforce

6. Security testing

   An additional training topic that is often overlooked is security testing. Security testing practices in far too many of today's software development organizations consists of little more than a late-cycle penetration test. As we've seen in Adapting Penetration Testing for Software Development Purposes[95], this approach is inadequate. One step in adopting better testing practices is to train the development and testing team on how to do security testing in depth [Howard 2006[96]].

   1. Fuzz testing
   2. Penetration testing
   3. Run-time verification

Of course, the above outlines are quite simplistic and generic views of the topics to be covered. Additional examples of course are cited in the list at the end of this article, as well as in [Howard 2006[98]]. Additionally, some worthwhile considerations in creating or selecting appropriate course material might include the following:

- Beyond the basics, courses should be as specific to the development environments in use as possible.
- Courses should include hands-on exercises/labs whenever feasible.
- Course material should cite common defects with specific code examples.
- Secure coding guidance should be prescriptive with ample code pattern examples for the students to study from.
- Instructors with exceptional communication skills are vital, but also look for instructors with hands-on software development experience.

Once a firm conceptual foundation has been laid for the students, a library or repository of up-to-date reference information should be made readily available to them. This should include external sources of information such as books and published papers, as well as internal sources such as (security vetted) design architectures, design documents, and source libraries.

Measuring Knowledge

A serious training initiative should take steps to verify and validate the students' knowledge base, but many questions quickly emerge [Howard 2006[102]]. Several software security certification programs are beginning to emerge in the commercial marketplace, which might help address this issue in the future. At a minimum, however, development organizations should mandate training attendance and record employee participation. If the organization tracks other related security metrics (e.g., code defect density per thousand lines of code), it may consider trying to correlate course attendance with defect density, but that is a degree of measurement that few organizations can achieve.

Business Case

As the practice of software security catches on and grows throughout the software development community, training and awareness initiatives are vital to adoption among developers and managers alike. This is particularly the case as few (if any) professional software developers today have undergone anything more than rudimentary on-the-job exposure to software security issues, much less anything in the form of academic instruction.

98. #dsy256-BSI_Howard
102. #dsy256-BSI_Howard

For software security best practices to be successfully adopted in industry, there must be senior-level buy-in. This can be accomplished in a number of ways, including a clear and concise awareness training program that presents senior decision makers with the issues and tradeoffs involved in delivering secure software.

Further, mid-level engineering management needs to be aware not just of the issues associated with delivering secure software but with the software security best practices that they should be incorporating into their groups' development processes and methodologies.

Lastly, software developers themselves, from architects and designers through coders and testers, need to be thoroughly trained in all of the above, plus all of the technology specifics involved in designing, coding, and testing software in the technologies that they work with.

Without these things, it is highly unlikely that software security initiatives can succeed in a substantial way. Trying to accomplish a software security agenda from a "grass roots" or "bottom up" perspective is not likely to accomplish more than superficial change.

Glossary

incident	Any real or suspected adverse event in relation to the security of computer systems or computer networks.

Commercial Training Examples

Aspect Security, Inc., Application Security Education and Training[111]

McAfee, Inc., Foundstone Training[112]

KRvW Associates, LLC., Training Services[113]

LogiGear, Inc., Web and Software Application Security Testing[114]

Microsoft Corp., Clinic 2806: Microsoft® Security Guidance Training for Developers[115] (and other courses)

Netcraft, Inc., Web Application Security Course[116]

Next Generation Security Software, Ltd., Security Training[117]

Outbreak Security, LLC., Course Offerings[118]

Paladion Networks Pvt. Ltd., Training for Software Developers[119]

The SANS Institute, Inc.[120]

Security Innovation, Inc., Application Security Education[121]

Symantec Corp., Application Security Principles and Security in Software Development Lifecycle[122]

Vigilar Corp., Writing Secure Code[123]

111. http://www.aspectsecurity.com/training.htm
112. http://www.foundstone.com/us/education-overview.asp
113. http://www.krvw.com/training/training.html
114. http://www.logigear.com/training/course_catalog/course.asp?courseId=20
115. https://www.microsoftelearning.com/eLearning/courseDetail.aspx?courseId=26043
116. http://audited.netcraft.com/web-application-course
117. http://www.ngssoftware.com/consulting/training/
118. http://www.outbreaksecurity.com/courses/courses.html
119. http://www.paladion.net/paladion.php?id=18
120. https://www.sans.org/
121. http://www.securityinnovation.com/services/education/index.shtml
122. https://education.symantec.com/Saba/Web/sena
123. http://www.vigilar.com/training/

Academic Curricula *Sampling*

Carnegie Mellon University CS curriculum, http://www.csd.cs.cmu.edu/education/bscs/index.html#curriculum

University of California at Davis CS curriculum, http://www.cs.ucdavis.edu/courses/exp_course_desc/index.html

George Washington University CS curriculum, http://www.cs.gwu.edu/news_events/program_news/newcourses.html

Massachusetts Institute of Technology EECS Undergraduate Program, http://www.eecs.mit.edu/ug/index.html

Stanford University CS curriculum, http://cs.stanford.edu/Courses/

University of Virginia CS curriculum, http://www.cs.virginia.edu/classes/index.php

Virginia Tech CS curriculum, http://www.cs.vt.edu/home/courses.html

References

[Howard 2006]

Howard, M.; Lipner, S. *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*. Redmond, WA: Microsoft Press, 2006 (ISBN 977-07356-2214-2).

Carnegie Mellon Copyright

1. mailto:permission@sei.cmu.edu